ELSEVIER

Artificial Intelligence 137 (2002) 197–216

Artificial Intelligence

www.elsevier.com/locate/artint

A re-weighting strategy for improving margins ☆

Fabio Aiolli, Alessandro Sperduti *

University of Pisa, Department of Computer Science, Corso Italia 40, 56125 Pisa, Italy

Received 9 July 2001

Abstract

We present a simple general scheme for improving margins that is inspired on well known margin theory principles. The scheme is based on a sample re-weighting strategy. The very basic idea is in fact to add to the training set new replicas of samples which are not classified with a sufficient margin.

As a study case, we present a new algorithm, namely TVQ, which is an instance of the proposed scheme and involves a tangent distance based 1-NN classifier implementing a sort of quantization of the tangent distance prototypes. The tangent distance models created in this way have shown a significant improvement in generalization power with respect to standard tangent models. Moreover, the obtained models were able to outperform other state of the art algorithms, such as SVM, in an OCR task. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Tangent distance; Margins; Re-weighting; Learning Vector Quantization; Nearest Neighbor; Multi-class classification; Invariant pattern recognition; Machine learning

1. Introduction

In this paper we introduce a simple additive re-weighting method that is able to improve the margin distribution on the training set. Recent results in computational learning theory [1,8,16] have tightly linked the expected risk of a classifier (i.e., the probability of misclassification of a pattern drawn from an independent random distribution $\mathcal{D}$), with the distribution of the margins μ_i for the examples in a given training set S. In particular, given a value $\theta > 0$, there exist upper bounds that, with high probability, limit the expectation of

☆ This is an extended version of the paper presented at IJCAI-2001, Seattle, WA, USA.

* Corresponding author.

E-mail addresses: aiolli@di.unipi.it (F. Aiolli), perso@di.unipi.it (A. Sperduti).

the generalization error of a general hypothesis h, depending on how many examples have margin less than the threshold value θ. Such bounds have the form

$$P_{\mathcal{D}}(\mu \leqslant 0) \leqslant P_S(\mu \leqslant \theta) + \widetilde{\mathrm{O}}\left(\sqrt{\frac{d}{\theta^2|S|}}\right),$$

where d is the VC-dimension of the hypothesis space. From a general analysis of these bounds it results that we can expect better performance on generalization (minimal error on test data) when most of the patterns have high margins and when large values to the θ parameter can be assigned in the formula above.

The aforementioned bounds are at the basis of the theory of two of the most impressive algorithms: Support Vector Machines and Boosting. Both SVM's and Boosting's effectiveness is largely due to the fact that they, directly or indirectly, effectively improve the margins on the training set. In particular, SVM explicitly finds the hyperplane with the largest minimum margin in a dimensional-augmented space where training points are mapped by a kernel function. In this case, margin theory permits to explain impressive performance even in very high dimensional spaces, where the "curse of dimensionality" is expected to reduce the probability to get good performance. Most of the recent efforts in SVMs have been in the choice of suitable kernels for particular applications. For example, in OCR problems, the polynomial kernel was proven to be very effective.

On the other side, boosting algorithms, and in particular the most famous version AdaBoost, produce weighted ensemble of hypotheses, each one trained in such a way as to minimize the empirical error in a given "difficult" distribution of the training set. Again, it has been shown [7] that boosting essentially is a procedure for finding a linear combination of weak hypotheses which minimizes a particular loss function dependent on the margins on the training set, literally $L = \sum_i \exp(-\mu_i)$. Recently, research efforts related to boosting algorithms faced the direct optimization of the margins on the training set. For example, this has been done by defining new margin-based cost functions and searching for combinations of weak hypotheses so to minimize these functions [6].

We will follow a related approach that aims to find a single (eventually non-linear) optimal hypothesis where the optimality is defined in terms of a loss-function that is dependent on the distribution of the margins on the training set. This function will induce greater losses for patterns that do not reach a predefined threshold on the margins in such a way to constrain the decision function just to improve the performance on those patterns. In order to minimize this loss we propose a re-weighting algorithm that maintains a set of weights associated with the patterns in the training set. The weight associated to a pattern is iteratively augmented when the margin of the current hypothesis is below the threshold. In this way a new distribution on the training data is induced. Furthermore, a new hypothesis is then computed that improves the expectation of the margin on the new distribution. In the following we prove that, if it is not possible to have all the patterns with margin above the threshold, the distribution converges to a uniform distribution on the patterns of the training set for which the margin cannot be increased above the threshold.

The simple scheme described above has been applied to an OCR pattern recognition problem, where the classification is based on a 1-NN tangent distance classifier [11], obtaining a significant improvement in generalization with respect to previous tangent distance based algorithms. Many tangent-distance based schemes have been applied recently with good results to OCR problems. These techniques have been shown suited when invariances to a priori known input transformations are beneficial.

The proposed new algorithm builds a set of models for each class in a way that recall the learning of codebook vectors in the Learning Vector Quantization procedure (LVQ [5]). In our case, the LVQ algorithm is extended to tangent distance models and re-weighting. In the following we will refer to this new algorithm as Tangent Vector Quantization (TVQ).

The paper is organized as follows. In Section 2, we introduce the concept of margin regularization via the input distribution on the training set. Specifically, we present the θ-Margin Re-weighting Strategy, which holds the property to guarantee the convergence of the input distribution. In Section 3 we introduce the tangent distance models and in Section 4 we introduce a general definition for the margins in a 1-NN scheme that considers the discriminative ratio observed for a particular pattern. In Section 5 we define the TVQ algorithm as an instance of the scheme applied to tangent distance based classifiers. Finally, in Section 6 we present empirical results comparing TVQ with other 1-NN based algorithms, including SVM.

2. Margin regularization

When learning takes place, the examples tend to influence in different ways the discriminant function of a classifier. A discriminant function can be viewed as a resource that has to be shared among different clients (the examples). Often, when pure Empirical Risk Minimization (ERM) principle is applied, that resource is used in a wrong way since, typically, it is almost entirely used by a fraction of the training set. Margin theory formally tells us that it is preferable to regularize the discriminant function in such a way to make the examples sharing more equally its support.

Inspired on the basic ideas of margin optimization, here, we propose a simple procedure applicable, eventually, to any ERM based algorithm. It permits to regularize the parameters of a discriminant function so to obtain hypotheses with large margins for many examples in the training set.

Without generality loss we consider the margin for a training example as a real number, taking values in $[-1,+1]$, representing a measure of the confidence shown by a classifier in the prediction of the correct label. In a binary classifier, e.g., the perceptron, the margin is usually defined as $yf(x)$ where $y \in \{-1,+1\}$ is the target and $f(x)$ is the predicted output computed by the classifier eventually re-conduced to the $[-1,+1]$ range by a monotonic (linear or sigmoidal) transformation of the output. Anyway, we assume that a positive value of the margin will correspond to a correct classification of the example.

Given a binary classification problem, in general, we are interested in minimizing the 0–1 loss function:

$$Loss_{0-1}\big(y, f(x)\big) = \begin{cases} 0 & \text{if } yf(x) \geqslant 0, \\ 1 & \text{if } yf(x) < 0, \end{cases} \tag{1}$$

which, however, is difficult to minimize due to its discrete nature. A typical approach to face this problem is to define an alternative loss which constitutes an upper bound to the 0–1 loss function easier to work with. We follow this approach. Specifically, given the function $\mu_h(x_i)$ that, provided an hypothesis h, associates to each pattern its margin, we would like to define a loss function that, when minimized, will permit to obtain hypotheses with large margins (greater than a fixed threshold θ) for many examples in the training set. For this, we propose to minimize a (relaxed) function that is similar to the SVM's slack variables loss function

$$L = \sum_{x_i \in S} \big(\theta - \mu_h(x_i)\big)[\![\mu_h(x_i) < \theta]\!], \tag{2}$$

where S is a training set with N examples, and $[\![\mu_h(x_i) < \theta]\!] = 1$ if $\mu_h(x_i) < \theta$, and 0 otherwise. A term in the function L is null for margins higher than the threshold θ and is linear with respect to the values of the margins when they are below the margin threshold.

Since the direct optimization of this functional is difficult, we suggest to minimize L indirectly by means of a two-step iterative method that "simultaneously" (1) searches for an a priori distribution $\{\gamma_i\}$ for the examples that, given the current hypothesis h, better approximates the function $[\![\mu_h(x) < \theta]\!]$ and (2) searches for a hypothesis h (e.g., by a gradient based procedure) that, provided the distribution $\{\gamma_i\}$, improves the weighted function

$$H = \sum_{p=1}^{N} \gamma_p \mu_h(x_p). \tag{3}$$

The maximization of the new given formulation in Eq. (3) is equivalent to the minimization of that given in Eq. (2) provided that $\{\gamma_i\}$ converges to the uniform distribution on the *θ-mistakes* (patterns that have margin below the threshold). In this way we easily re-conduct a problem of margin optimization to one that is completely based on a weighted empirical loss.

2.1. The θ-Margin Re-weighting Strategy

The proposed algorithm consists of a series of trials. An empirical optimization process, that explicitly maximizes the function H according to the current distribution for the examples, works on an artificial training set S', initialized to be equal to the original training set S. For each step t, $k(t)$ replicas of those patterns in S that have margin below the fixed threshold θ are added to S' augmenting their density in S' and consequently their contribution in the optimization process. The algorithm is shown in Fig. 1 where $w_i(t)$ denotes the number of occurrences in the augmented training set S' of the pattern x_i.

Just to start we can easily show nice statistical properties of the algorithm which will turn useful when initializing models with different (randomly generated) class-conditional distributions of the patterns of the same class (see Section 5.1). First, we can show that, at each step, it is possible to express the mean and the covariance matrix of the examples in the augmented training set S' in closed form as weighted combinations of the original components

$$C_{S'} = \sum_{j=1}^{|S|} \gamma_j x_j \quad \text{and} \quad \Sigma_{S'} = \sum_{j=1}^{|S|} \gamma_j (x_j - C_{S'})(x_j - C_{S'})^{\mathrm{t}}. \tag{4}$$

```
Input:
  T: number of iterations;
  $\mathcal{H}$: hypothesis space;
  $\theta$: margin threshold;
  $k(\cdot) \in (0, K]$: bounded function;
  $S = \{(x_i, y_i)\}_{i=1,\dots,N}$: training set;
Initialize
  $h_0 \in \mathcal{H}$ (initial hypothesis);
  for $i = 1, \dots, N$
    $w_i(1) \leftarrow 1$, $\gamma_i(1) \leftarrow \frac{1}{N}$;
for $t = 1, \dots, T$
  begin
    find $h_t$ such that
      $\sum_{i=1}^{N} \gamma_i(t)\mu_i(t) > \sum_{i=1}^{N} \gamma_i(t)\mu_i(t-1)$;
    $w_i(t+1) \leftarrow w_i(t) + k(t)[\![\mu_i(t) < \theta]\!]$;
    $\gamma_i(t+1) \leftarrow \frac{w_i(t+1)}{\sum_{j=1}^{N} w_j(t+1)}$;
  end
return $h_T$;
```

Fig. 1. The θ-Margin Re-weighting Strategy.

Proof. Suppose to have a set of size N with (possibly replicated) elements of the set S having n_j occurrences of patterns x_j and suppose for every j to add k_j replicas of patterns x_j to this set. Since there are k_j new copies of the same pattern x_j, the new mean will be therefore computed as

$$C = \frac{1}{N + \sum_j k_j}\left(\sum_j n_j x_j + \sum_j k_j x_j\right) = \sum_j \gamma_j x_j$$

where we simply set $\gamma_j = (n_j + k_j)/(N + \sum_j k_j)$. Moreover, it is easy to verify that for the new induced distribution γ, $\sum_j \gamma_j = 1$ holds.

Similarly, the corresponding covariance matrix will be defined as

$$\begin{aligned}\Sigma &= \frac{1}{N + \sum_j k_j}\left(\sum_j n_j (x_j - C)(x_j - C)^{\mathrm{t}} + \sum_j k_j (x_j - C)(x_j - C)^{\mathrm{t}}\right) \\ &= \sum_j \gamma_j (x_j - C)(x_j - C)^{\mathrm{t}}.\end{aligned}$$

This more general result can be easily instantiated to the case of the algorithm by setting $N = |S|$, $n_j = 1$ for all j and by setting k_j to the cumulative number of patterns x_j that we have added to the augmented training set up to a certain step. □

In the following section, we will prove that, if it is not possible to empty the set of θ-mistakes, the θ-Margin Re-weighting Strategy described above makes the distribution γ approach a uniform distribution on the θ-mistakes, provided that $k(t)$ is bounded, independently from the hypothesis space considered.

2.2. Convergence of the distribution

For each trial t, given the margin of each example in the training set S, we can partition the training sample as $S = S_M(t) \cup S_C(t)$ where $S_M(t)$ is the set of θ-mistakes and $S_C(t) = S - S_M(t)$ is the complementary set of θ-correct patterns.

Let denote $W(t) = |S'(t)|$ and let $w_i(t)$ be the number of occurrences of pattern x_i in S' at time t, with density $\gamma_i(t) = w_i(t)/W(t)$. Moreover, let $\Lambda(t)$ be a suitable function of t such that $W(t+1) = W(t)\Lambda(t)$.

Let $w_i(t+1) = w_i(t) + k(t)$ be the update rule for the number of occurrences in S', where $k(t)$ is bounded and takes values in $(0, K]$ (note that $k(t)$ may change at different iterations but it is independent from i). It's easy to verify that $\Lambda(t) \geqslant 1$ for each t because of the monotonicity of $W(t)$, and that $\Lambda(t) \to 1$ with the number of iterations. In fact

$$1 \leqslant \frac{W(t+1)}{W(t)} = \frac{W(t) + \Delta W(t)}{W(t)} \leqslant 1 + \frac{K|S|}{W(t)} \to 1.$$

At time $t+1$ we have

$$\gamma_i(t+1) = \frac{w_i(t) + k(t)[\![\mu_i(t) < \theta]\!]}{\Lambda(t)W(t)} = \frac{\gamma_i(t) + \frac{k(t)}{W(t)}[\![\mu_i(t) < \theta]\!]}{\Lambda(t)}.$$

First of all we show that the distribution converges. This can be shown by demonstrating that the changes tend to zero with the number of iterations, i.e., $\forall x_i \in S$, $|\Delta\gamma_i(t)| \to 0$.

We have

$$\Delta\gamma_i(t) = \frac{\frac{k(t)}{W(t)}[\![\mu_i(t) < \theta]\!] - (\Lambda(t) - 1)\gamma_i(t)}{\Lambda(t)},$$

which can be easily bounded in module by a quantity that tends to 0:

$$|\Delta\gamma_i(t)| \leqslant \frac{\frac{K}{W(t)} + (\Lambda(t) - 1)\gamma_i(t)}{\Lambda(t)} \to 0.$$

We now show to which values they converge. Let m_i and ε_i be, respectively, the cumulative number and the mean ratio of θ-mistakes for x_i on the first t epochs, $\varepsilon_i = m_i(t)/t$, then

$$\begin{aligned}\gamma_i(t) &= \frac{w_i(0) + E[k(t)]m_i(t)}{W(0) + E[k(t)]\sum_{j=1}^{N} m_j(t)}\\ &= \frac{1 + tE[k(t)]\varepsilon_i}{N + tE[k(t)]\sum_{j=1}^{N}\varepsilon_j} \to \frac{\varepsilon_i}{\sum_{j=1}^{N}\varepsilon_j}.\end{aligned}$$

Given the convergence of the optimization process that maximizes H in Eq. (3), the two sets S_M and S_C are going to become stable and the distribution on S will tend to a uniform distribution in S_M (where $\varepsilon_i \to 1$) and will be null elsewhere (where $\varepsilon_i \to 0$). This can be understood in the following way as well.

Given the definition of the changes made on the gamma values on each iteration of the algorithm, we calculate the function that we indeed minimize. Since $\Delta w(t) = k(t) \cdot [\![\mu(t) < \theta]\!]$, after some algebra, we can rewrite $\Delta\gamma_i(t)$ as:

$$\Delta\gamma_i(t) = \frac{\Delta W(t)}{W(t+1)}\left(\frac{[\![\mu_i(t) < \theta]\!]}{|S_M(t)|} - \gamma_i(t)\right).$$

Thus, assuming $\Delta W(t) \neq 0$, $\Delta\gamma_i(t) = 0$ when $\gamma_i = (1/|S_M(t)|)[\![\mu_i(t) < \theta]\!]$, for which the minimum of function

$$E_\gamma = \frac{\Delta W(t)}{W(t+1)}\left(\frac{1}{2}\sum_i \gamma_i^2(t) - \frac{1}{|S_M(t)|}\sum_i \gamma_i(t)[\![\mu_i(t) < \theta]\!]\right)$$

is reached. Note that, the minimum of E_γ is consistent with the constraint $\sum_i \gamma_i = 1$. This energy function is modulated by a term decreasing with the number of iterations, dependent on the $k(t)$ used but independent from gamma, that can be considered as a sort of annealing introduced in the process. This becomes evident in the specific case where $k(t) = |S|/|S_M(t)|$. In this case E_γ is further simplified in:

$$\frac{1}{t+1}\left(\frac{1}{2}\sum_i \gamma_i^2(t) - \frac{1}{|S_M(t)|}\sum_i \gamma_i(t)[\![\mu_i(t) < \theta]\!]\right).$$

Finally, it is remarkable that the uniform distribution over the θ-mistakes S_M which minimizes the function E_γ, also maximizes the entropy over the θ-mistakes S_M, i.e.,

$$\min_\gamma E_\gamma = \min_\gamma \sum_{i \in S_M} \gamma_i \log(\gamma_i) = \max_{\gamma, S_M} \mathit{Entropy}(\gamma). \tag{5}$$

Thus, the distribution returned by the proposed procedure is actually making the *less restrictive* hypotheses over the parameters, so as prescribed by the Maximum Entropy Principle and the Occam's Razor.

In the following, we study a specific instance of the θ-Margin Re-weighting Strategy applied to tangent distance based classifiers.

3. Tangent distance models

In this paper, we are particularly interested in distances that are invariant to given transformations. Invariance to given transformations is very important in classification tasks. For example, when recognizing handwritten characters, it would be desirable to obtain the correct classification even if the character is slightly rotated, or translated, or even stretched. This problem has been faced with the introduction of Tangent Distance [11], which basically computes an approximation of the minimum distance between the possible transformations that two patterns to be compared can undertake. Given a set of m transformations parametrized through the vector α, the 'two-sided' tangent distance between two patterns x and y is defined by

$$d_{T_2}(x, y) = \min_{\alpha_x, \alpha_y} \left\| \tilde{x}(\alpha_x) - \tilde{y}(\alpha_y) \right\|, \tag{6}$$

where $\tilde{x}(\alpha_x)$ and $\tilde{y}(\alpha_y)$ are linear approximations of the manifolds induced by transforming the patterns according to the given transformations.

Unfortunately, tangent distance computation is time consuming. This may be a problem when the distance is calculated many times (as in the k-NN algorithm). Different solutions have been proposed to reduce the number of distances computed [3,9,10,12,13,15]. Among these, the algorithm proposed by Hastie et al. [3] for the generation of prototype models that represent an entire set (class) of patterns, is the most interesting. Specifically, a prototype model $M(\alpha)$ is constituted by a centroid and a set of tangent vectors combined via parameters α:

$$M(\alpha) = C + \sum_{k=1}^{m} T_k \alpha_k,$$

where C is the *centroid* and the set $\{T_k\}$, $k = 1, \ldots, m$, constitutes the basis of the associated invariant subspace of dimension m. The model represents either the mean pattern of the class, via its centroid, and the principal transformations that the patterns can undertake in the training set, via the linear tangent subspace. This kind of models are descriptive[1] and typically used within a k-NN scheme.

Given a class y, Hastie et al. [3] suggest to select the centroid C and the tangent subspace $\{T_k\}$ as the minimizer of the error function

$$\sum_{i=1}^{N_y} d_{T_2}(x_i, M) = \sum_{i=1}^{N_y} \min_{\alpha_x, \alpha_M} \left\| x_i(\alpha_x) - M(\alpha_M) \right\|^2. \tag{7}$$

The above definition constitutes a difficult optimization problem, which however can be solved for a fixed value of m (i.e., the invariant subspace dimension) by an iterative algorithm based on Singular Value Decomposition [3]. In the following we refer to this algorithm with the acronym HSS 2-sided.

The computational burden due to tangent distance can be reduced by resorting to the *one-sided tangent distance* [9], which computes the distance between a pattern and a subspace in the following way

$$d_{T_1}(x, y) = \min_{\alpha} \left\| x - \tilde{y}(\alpha) \right\|. \tag{8}$$

In the following, we only consider the 1-sided version of tangent distance between patterns and tangent models, denoting it with $d_T(x, M)$.

Taking into account the tangent distance formulation as given in Eq. (8) and assuming with no loss in generality that the tangent vectors are ortho-normal, it is quite easy to verify that the squared tangent distance between a pattern x and a model M, $M = (C, \{T_k\})$, can be written as

$$z = d_T^2(x, M) = \delta^{\mathrm{t}} \delta - \sum_{k=1}^{m} \alpha_k^2, \tag{9}$$

where $\delta = x - C$, $\alpha_k = \delta^{\mathrm{t}} T_k$ and δ^{t} denotes the transpose of δ.

[1] See [4] for a discussion about the advantages of using descriptive models.

Furthermore, given a set $\{x_p\}$ of positive instances of class y, the optimal non-discriminant model M_y is the solution of the minimum problem

$$M_y = \arg\min_M \sum_{p=1}^{N_y} d_T^2(x_p, M) = \arg\min_M \sum_{p=1}^{N_y} \left(\delta_p^{\mathrm{t}} \delta_p - \sum_{k=1}^{m} \alpha_{p,k}^2 \right), \tag{10}$$

where N_y is the cardinality of the subset of positive instances in the training set. This problem can be solved resorting to principal component analysis theory, also called *Karhunen–Loéve Expansion*. In fact, Eq. (10) can be minimized by choosing the centroid C as the average over all available training samples x_p of class y, and T_k's as the most representative eigenvectors (principal components) of the covariance matrix Σ, where

$$\Sigma = \frac{1}{N_y} \sum_{p=1}^{N_y} (x_p - C)(x_p - C)^{\mathrm{t}}.$$

In the rest of the paper we will refer to the above algorithm as HSS 1-sided or simply HSS. It must be observed that, by construction, the HSS algorithms, both 1-sided and 2-sided, return non-discriminant models. In fact, they use only the evidence provided by positive examples of the target class.

Anyway, a discriminant model can always be generated by using the *TD-Neuron* [14], a constructive algorithm based on gradient descent and the one-sided version of the tangent distance.

Another family of well-performing algorithms is Learning Vector Quantization (LVQ) [2]. Different versions of this strategy are available, each one differing slightly from the others. In the simplest and more general case these algorithms quantize input patterns into codebook vectors c_i and use these vectors for 1-NN classification. Several codebooks may correspond to a single class. At each step of the codebook learning, for each input pattern x_i the algorithm finds the element c_k closest to x_i. If c_k is associated with a different class from that of x_i then c_k is updated by $c_k \leftarrow c_k - \eta(t)(x_i - c_k)$.

Empirical results over the NIST-3 database showed that the TD-Neuron, is superior to both SVD and LVQ based algorithms, since it reaches a better trade-off between error and rejection.

With the contribution of this paper, we propose an algorithm that, in addition to be discriminative, is able to control the margins of tangent distance based classification with the explicit goal of minimizing the generalization error.

4. Margins in a 1-NN framework

Given a training example $(x_i, y_i) \in S$, $y_i \in Y$, and a fixed number of models for each class, we give a definition of the margin for an example when classified by a distance based 1-NN classifier.

Given a sample (x_i, y_i), let z_i^p and z_i^n be the squared distances between the nearest of the positive set of models and the nearest of the negative sets of models, respectively. We can define the margin of a pattern in the training set as:

$$\mu_i = \frac{z_i^n - z_i^p}{z_i^n + z_i^p}. \tag{11}$$

This formula takes values in the interval $[-1, +1]$ representing the confidence in the prediction of the 1-NN classifier. Higher values of the μ_i's can also be viewed as an indication of a higher discriminative power of the set of models with respect to the pattern. Moreover, a pattern will result correctly classified in the 1-NN scheme if and only if its margin is greater than zero.

4.1. *Improving hypotheses by gradient search*

Given the above definition of margins for a general distance based nearest neighbor classifier and given a differentiable distance definition, we can see that the choice of the new hypothesis in the θ-Margin Re-weighting Strategy can be implemented as a single step of the gradient ascent algorithm applied to the margins on the current input distribution. In our case, considering the tangent distance formulation as given in Eq. (9) we can verify that it is completely defined by scalar products. So, we can derivate it with respect to the centroid C and the tangent vectors $\mathcal{T} = \{T_k\}$ of the nearest positive model obtaining:

$$\frac{\delta z_i^p}{\delta C} = -2\left(\delta - \sum_{k=1}^{|\mathcal{T}|} \alpha_k T_k\right), \qquad \frac{\delta z_i^p}{\delta T_k} = -2\alpha_k \delta.$$

Considering that

$$\frac{\delta \mu_i}{\delta C} = \frac{\delta \mu_i}{\delta z_i^p} \frac{\delta z_i^p}{\delta C} \quad \text{and} \quad \frac{\delta \mu_i}{\delta T_k} = \frac{\delta \mu_i}{\delta z_i^p} \frac{\delta z_i^p}{\delta T_k}$$

we can compute the derivative of the margin (as defined in Eq. (11)) with respect to changes in the nearest positive model:

$$\begin{aligned}
\frac{\delta \mu_i}{\delta C} &= 4 \frac{z_i^n}{(z_i^n + z_i^p)^2} \left(\delta - \sum_{k=1}^{|\mathcal{T}|} \alpha_k T_k\right), \\
\frac{\delta \mu_i}{\delta T_k} &= 4 \frac{z_i^n}{(z_i^n + z_i^p)^2} \alpha_k \delta.
\end{aligned}$$

A similar solution is obtained for the nearest negative model since it only differs in changing the sign and in exchanging indexes n and p. Moreover, the derivatives are null for all the other models.

Thus, we can maximize the average margin in the training set if for each pattern presented to the classifier we move the nearest models in the direction suggested by the gradient. Note that, like in the LVQ algorithm, for each training example, only the nearest models are changed.

```
Input:
  $T$: number of iterations;
  $Q$: number of models per class;
  $\theta$: margin threshold;
Initialize
  $W(1) \leftarrow N, \gamma_i \leftarrow \frac{1}{N}$;
  $\forall y, q$, initialize $M_y^{(q)} \leftarrow (C_y^{(q)}, \mathcal{T}_y^{(q)})$ with random models;
for $t = 1, \ldots, T$
  $\forall y, \forall q, \Delta M_y^{(q)} = 0$;
  $\forall (x_i, y_i) \in S$, select $(q_p, \bar{y}_i, q_n)$ s.t. $M_{y_i}^{(q_p)}$ and $M_{\bar{y}_i}^{(q_n)}$ are the nearest, positive and negative,
  models. Compute $\mu_i(t)$ as in Eq. (11) and accumulate the changes on the nearest models
      $\Delta M_{y_i}^{(q_p)} \leftarrow \Delta M_{y_i}^{(q_p)} + \gamma_i(t) \frac{\delta \mu_i}{\delta M_{y_i}^{(q_p)}}$;
      $\Delta M_{\bar{y}_i}^{(q_n)} \leftarrow \Delta M_{\bar{y}_i}^{(q_n)} + \gamma_i(t) \frac{\delta \mu_i}{\delta M_{\bar{y}_i}^{(q_n)}}$;
  $\forall y, \forall q, M_y^{(q)} = M_y^{(q)} + \eta \Delta M_y^{(q)}$ and orthonormalize its tangents;
  $\forall (x_i, y_i) \in S$, update the distribution $\gamma_i$ by the rule
      $\gamma_i(t+1) \leftarrow \gamma_i(t) + \frac{[\![\mu_i(t) < \theta]\!]}{W(t)}$
  Normalize $\gamma_i$'s such that $\sum_{i=1}^{N} \gamma_i = 1$;
  $W(t+1) \leftarrow W(t) + |\{(x_i, y_i) | \mu_i(t) < \theta\}|$;
End
```

Fig. 2. The TVQ algorithm.

When maximizing the expected margin on the current distribution $\{\gamma_i\}$, i.e., H, for each model $M = (C, \{T_k\})$ we have:

$$\Delta C = \eta \sum_{i=1}^{|S|} \gamma_i \frac{\delta \mu_i}{\delta C}, \qquad \Delta T_k = \eta \sum_{i=1}^{|S|} \gamma_i \frac{\delta \mu_i}{\delta T_k},$$

where η is the usual learning rate parameter. In the algorithm (see Fig. 2), for brevity, we will group the above variations by referring to the whole model, i.e.,

$$\Delta M = \eta \sum_{i=1}^{|S|} \gamma_i \frac{\delta \mu_i}{\delta M}.$$

5. The TVQ algorithm

The algorithm (see Fig. 2) starts with random models and a uniform distribution on the training set. For each pattern, the variation on the closest positive and the closest negative models are computed accordingly to the density of that pattern on the training set S'. When all the patterns in S have been processed, the models are updated performing a step of a weighted gradient ascent on the mean value of the margin. Moreover, for each pattern in the training set such that the value of the margin is smaller than a fixed value, the distribution is augmented. The effect is to force the gradient ascent to concentrate on hardest examples in the training set. The increment to the distribution is justified in Section 2 and corresponds to the effect of adding a replica ($k(t) = 1$) of incorrectly classified patterns to the augmented training set.

5.1. Some comments on TVQ

The TVQ algorithm can be initialized in different ways. The first choice is to use randomly generated models. However, when the training set size is not prohibitive, we can drastically speed up the algorithm by taking as initial models the ones generated by any tangent distance based algorithm (e.g., HSS). However, in case of multiple models per class the initialization through the HSS method would generate identical models for each class and that would invalidate the procedure. A possible alternative choice in this case, that has been used in our implementation, is to generate HSS models on different (randomly generated) class-conditional distributions of the patterns of the same class. In order to do that, we have extended the basic HSS 1-sided algorithm to deal with non-uniform distributions of the patterns in a class. Basically, this has been done by using the generalized formulation for the covariance matrix of Eq. (4). Another solution, which is useful when the size of the training set is relatively large, is to initialize the centroids as the average of the positive instances and then generating random tangents.

Clearly, the speed of convergence with different initialization methods may be drastically different. This is largely due to the fact that when TVQ is initialized with HSS models it starts with a good approximation of the optimal hypothesis (see Fig. 3), while random initializations implicitly introduce an initial poor estimate of the final distribution due to the mistakes that most of the examples do on the first few iterations. Experimental results, however, have shown that the differences on the performance obtained by using different initialization criteria are negligible.

Simple minor variants to the algorithm (but not explored up to now) are possible. For example, multiple steps of gradient ascent can be made before updating the distribution and this can still improve the efficiency of the algorithm since, at each step, the algorithm gets a better estimate of the final mistake distribution.

6. Experiments

We compared the TVQ algorithm versus SVMs and other 1-NN based algorithms: HSS 1-sided, HSS 2-sided, TD-Neuron, and LVQ. The comparison was performed using exactly the same split of a dataset consisting of 10705 digits randomly taken from the NIST-3 dataset. The binary 128×128 digits were transformed into 64-grey level 16×16 images by

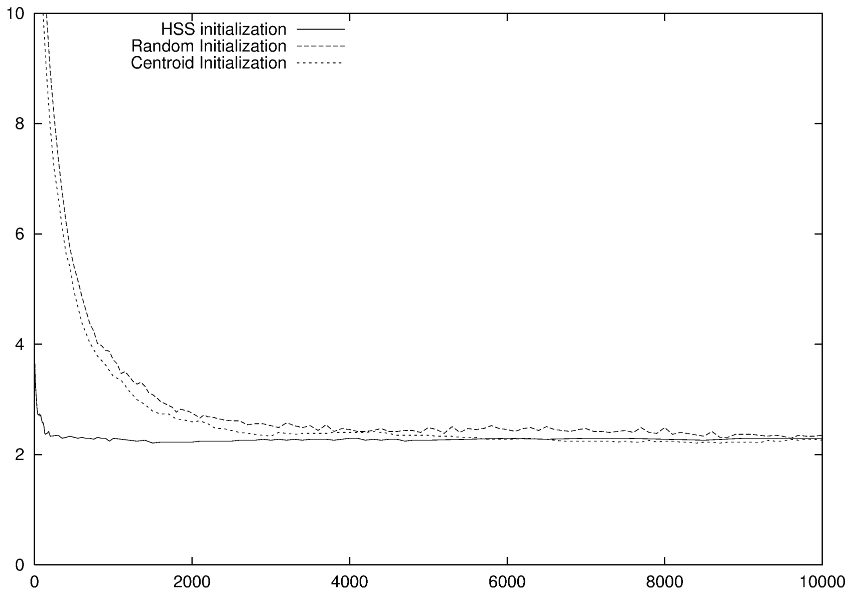


Fig. 3. Comparison among different initialization methods for the TVQ algorithm.

Table 1
Test results for different state of the art 1-NN methods

Method	Parameters for each class	Err%
HSS 1-sided	15 tangents	3.58
LVQ 2.1	16 codebooks	3.52
TD-Neuron	15 tangents	3.51
HSS 2-sided	9 tangents	3.40
Euclidean 1-NN	Training examples	3.16
SVM	Linear	10.64
SVM	Poly $d = 2$	2.82
SVM	Poly $d = 3$	3.23
SVM	Poly $d = 4$	4.02

a simple local counting procedure.[2] The only preprocessing performed was the elimination of empty borders. The training set consisted of 5000 randomly chosen digits, while the remaining digits were used in the test set.

The results for the test data obtained using state of the art algorithms are summarized in Table 1. For each algorithm, we reported the best result, without rejection, obtained for the dataset. Specifically, different LVQ algorithms were tested (optimized-learning-rate LVQ1, original LVQ1, LVQ2.1, LVQ3) using the LVQ PAK package [5]. However, we just report the results obtained by using LVQ2.1 with 1-NN based on Euclidean distance as

[2] The number of pixels with value equal to 1 is used as the grey value for the corresponding pixel in the new image.

Table 2
Test results for TVQ

	$\theta = 0.1$	$\theta = 0.3$		$\theta = 0.4$	
Q	$\lvert\mathcal{T}\rvert = 0$	$\lvert\mathcal{T}\rvert = 10$	$\lvert\mathcal{T}\rvert = 15$	$\lvert\mathcal{T}\rvert = 10$	$\lvert\mathcal{T}\rvert = 15$
1	6.40	2.40	2.20	3.00	2.22
3	4.26	2.10	2.26	2.43	2.10

classification rule, since this algorithm reached the best performance over an extended set of experiments, involving LVQ algorithms, with different settings for the learning parameters. Concerning the SVM training we used the SVM^{Light} package available on the Internet.[3] Different kernels were considered for the SVMs: linear and non-homogeneous polynomial with degrees 2, 3 and 4 (we used the default for the other parameters). Since SVMs are binary classifiers, we built 10 SVMs, one for each class against all the others, and we considered the overall prediction as the label with higher margin. The best performance has been obtained with a polynomial kernel of degree 2.

We ran the TVQ algorithm with two different values for θ and four different architectures. Moreover, we ran also two experiments just using centroids (i.e., $|\mathcal{T}_y| = 0$) with $\theta = 0.1$. The smaller value for θ has been chosen just to account for the far smaller complexity of the model.

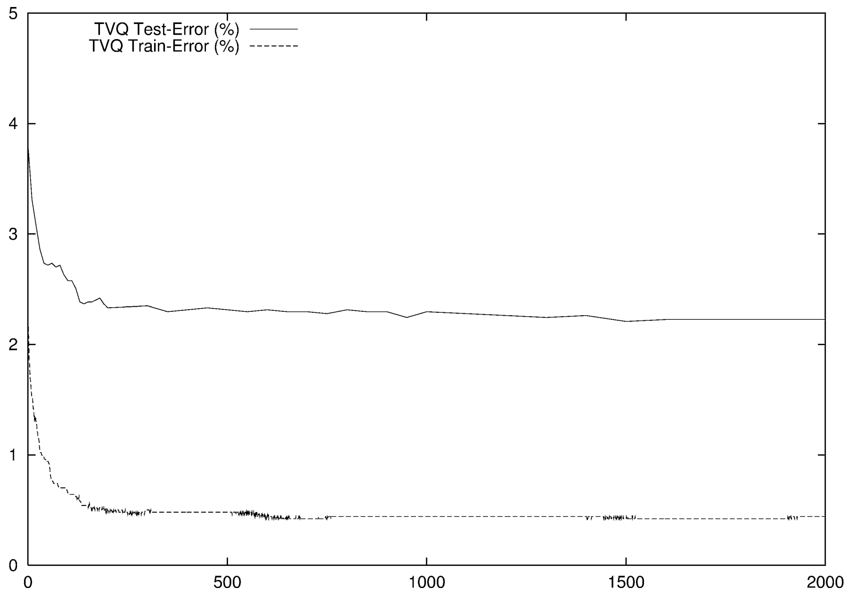
In almost all the experiments the TVQ algorithm obtained the best performance. Results on the test data are reported in Table 2. Specifically, the best result for SVM is worst than almost all the results obtained with TVQ. Particularly impressive are the results obtained by using only centroid vectors with no tangent space. From a computational point of view this corresponds to perform an LVQ (with re-weighting) with very few codebooks (one and three in our case) for each class.

In addition to better accuracy, TVQ returns far more compact models allowing a reduced response time in classification. In fact in the worst case the models returned by the TVQ involve a total of 480 vectors (one centroid plus 15 tangents for each one of the 30 models involved). The 1-NN using ten (one for each class) polynomial SVMs with $d = 2$, instead, needs 2853 support vectors in total, that become 1718 when considering only distinct support vectors.

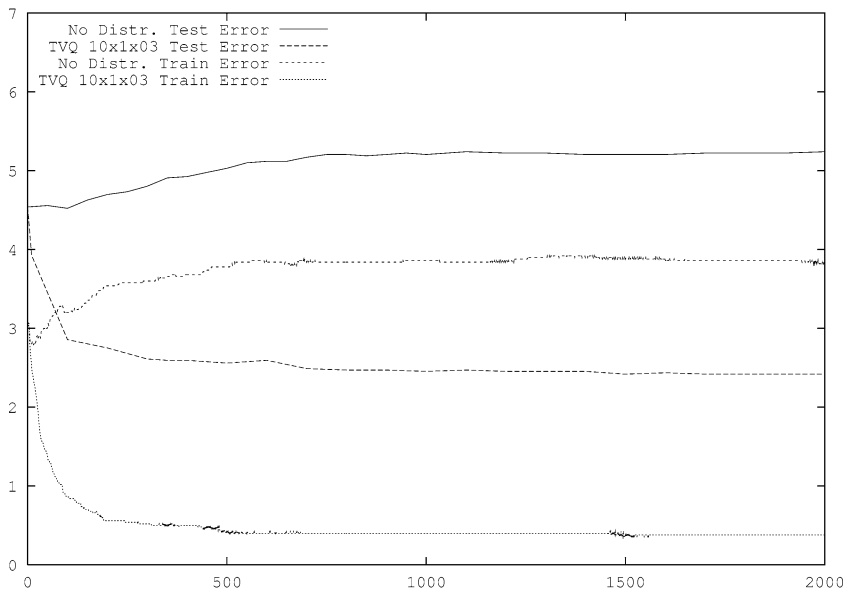
Typical error curves for the training and test errors of TVQ are reported in Fig. 4(a). From these plots it is easy to see that the TVQ doesn't show overfitting. This was also confirmed by the experiments involving models with higher complexity and moderately smaller values of θ.

The impact of margin regularization on the generalization error is shown in Fig. 4(b). In this plot train and test error curves of a session of the TVQ algorithm are compared with train and test error curves of models generated using exactly the same architecture but simply maximizing the mean of the margins without re-weighting. The plot makes clear that after very few iterations the simple maximization of the margins is actually worsening the basic performance obtained by the HSS model used for initialization.

[3] http://www-ai.cs.uni-dortmund.de/SOFTWARE/SVM_LIGHT/.



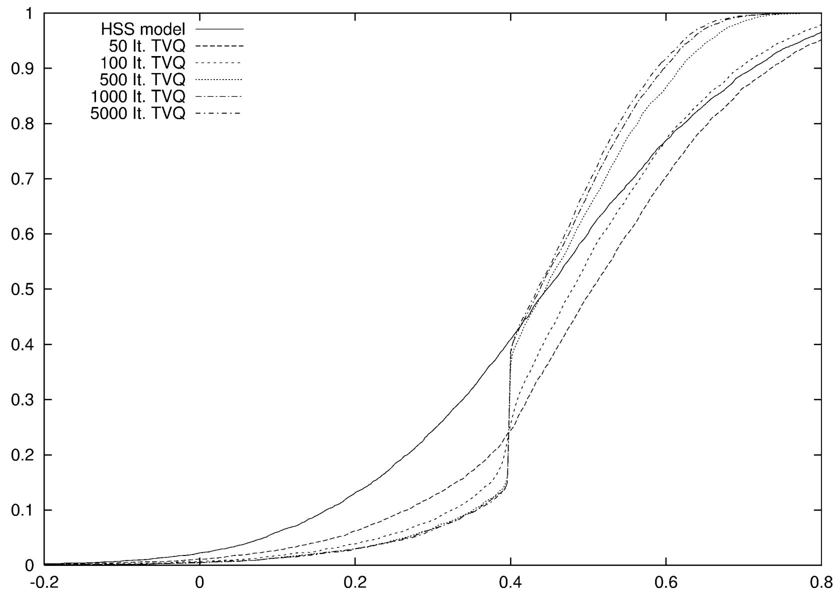
(a)



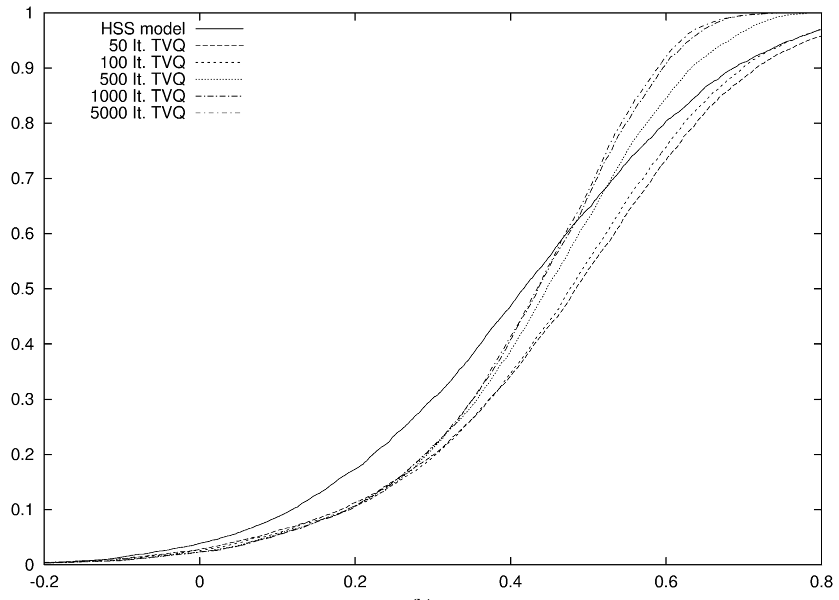
(b)

Fig. 4. Typical test and train error curves (a), and comparison of train and test error curves with and without re-weighting (b).

A visualization of the work done by the TVQ algorithm can be obtained by plotting the cumulative margin for the training and test sets (see Fig. 5). In these plots, the ratio of patterns having margins smaller than given values are considered. The impact of the



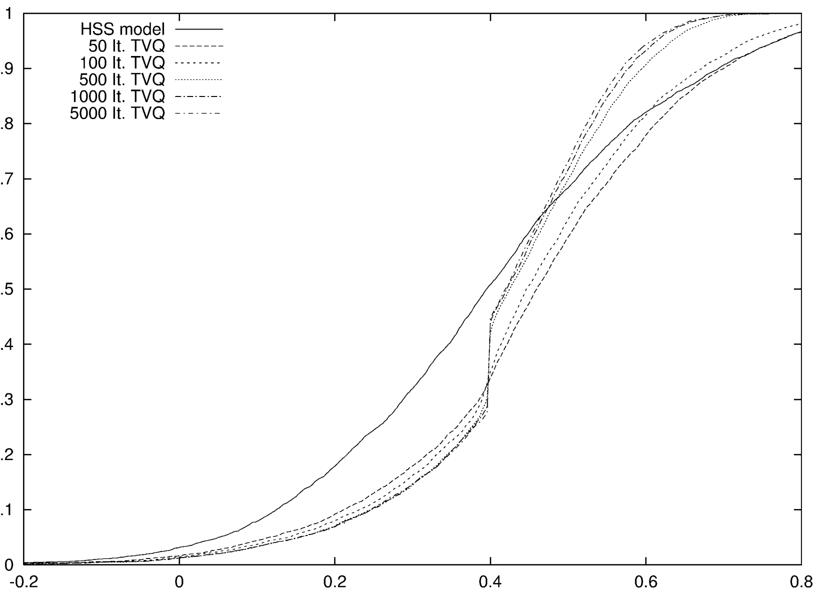
(a)



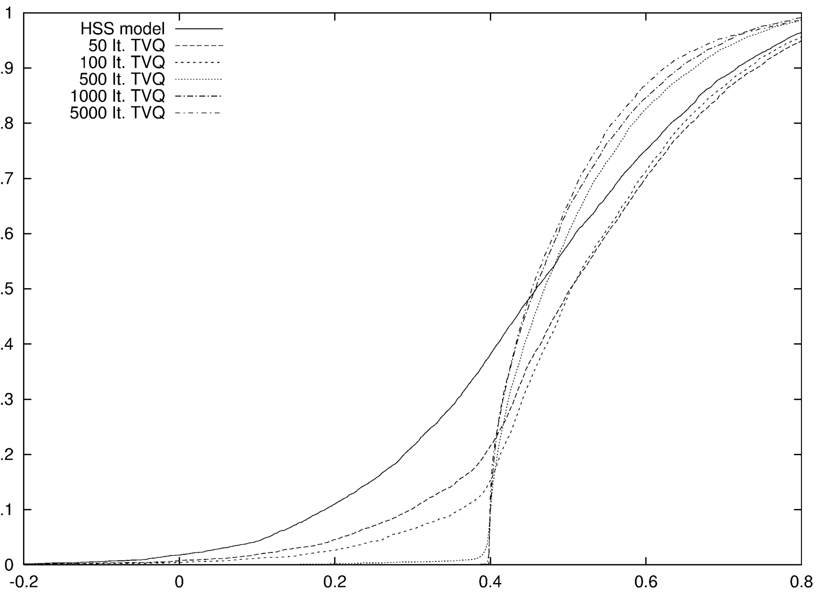
(b)

Fig. 5. Cumulative distribution of the margins on the training set (a) and on the test set (b) at different iterations of the TVQ algorithm (15×1 tangents, $\theta = 0.4$).

θ-margin on the final margin distribution on the training set is clearly shown in Fig. 5(a), where a steep increase of the distribution is observed in correspondence of θ at the expenses of higher values of margins. Even if at a minor extent, a similar impact on the margin



(a)



(b)

Fig. 6. Cumulative distribution of the margins for two extreme situations: low complexity model 10×1 (a) and high complexity model 15×3 (b) at different steps of the TVQ algorithm ($\theta = 0.4$).

distribution is observed for the test data in Fig. 5(b). Similar plots show how the complexity of the models involved in the TVQ learning strongly affects the layout of the cumulative margin (see Fig. 6). In particular in Fig. 6(a) the regularization effect is smaller since

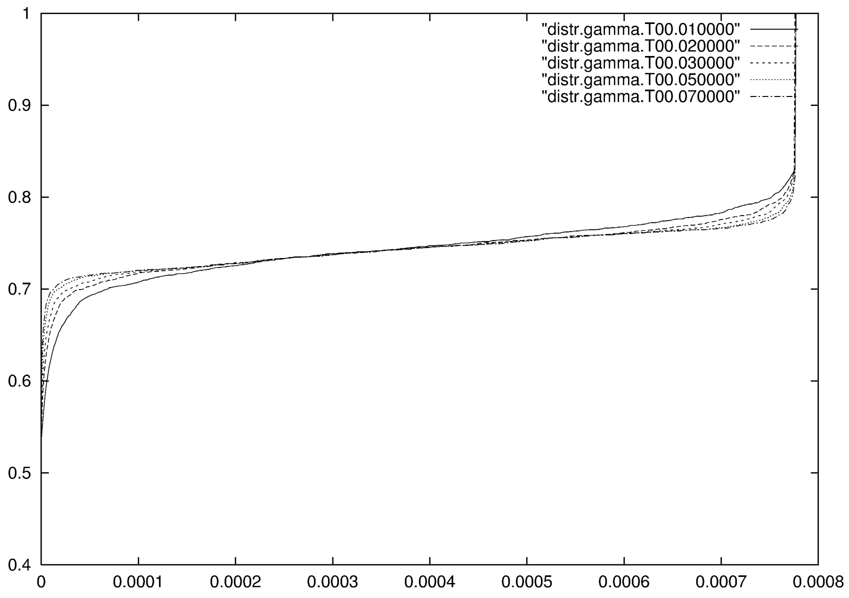


Fig. 7. Distribution layout at different iterations of the TVQ algorithm.

the architecture is not complex enough while in Fig. 6(b) the θ value is sufficiently low to permit the algorithm to eliminate all the θ-mistakes. This last situation could bring to overfitting even if this does not occur in this case. From our first analysis of the links between the layout of cumulative margin plots and the generalization error it seems that steeper layouts are preferable since they are associated to models which are both highly accurate and quite robust to overfitting.

We have experimentally verified that the gamma distribution converges to a uniform distribution over the θ-mistakes. This can be seen in Fig. 7, where the gamma distributions obtained at different iterations clearly show a convergence towards a uniform distribution on a sub-sample (in this case about 20%) of the training set (i.e., the θ-mistakes).

Finally, in Fig. 8 we have reported the rejection curves for the different algorithms. As expected, the TVQ algorithm was competitive with the best SVM, resulting to be the best algorithm for almost the whole error range.

7. Conclusions

We proposed a provably convergent re-weighting scheme for improving margins, which focuses on “difficult” examples. On the basis of this general approach, we defined a Vector Quantization algorithm based on tangent distance, which experimentally outperformed state of the art classifiers both in generalization and model compactness. These results confirm that the control of the shape of the margin distribution has a great effect on the generalization performance.

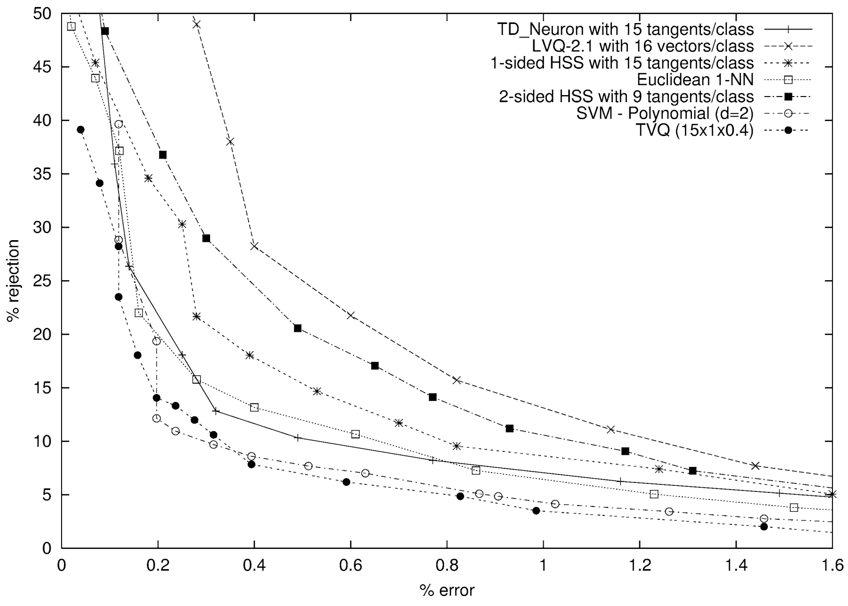


Fig. 8. Rejection curves for different 1-NN algorithms. The percentage of rejected patterns for each percentage of error is indicated. The rejection criterion is based on the difference between the two best predictions.

When comparing the proposed approach with SVM, we may observe that, while our approach shares with SVM the Statistical Learning Theory concept of uniform convergence of the empirical risk to the ideal risk, it exploits the input distribution to directly work on non-linear models instead of resorting to predefined kernels. This way to proceed is very similar to the approach adopted by Boosting algorithms. However, in Boosting algorithms, several hypotheses are generated and combined, while in our approach the focus is on a single hypothesis. This justifies the adoption of an additive re-weighting scheme, instead of a multiplicative scheme which is more appropriate for committee machines.

Acknowledgements

Fabio Aiolli wishes to thank Centro META—Consorzio Pisa Ricerche for supporting his PhD fellowship.

References

[1] P.L. Bartlett, The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network, IEEE Trans. Inform. Theory 44 (2) (1998) 525–536.
[2] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd ed., Wiley, New York, 2000.

[3] T. Hastie, P.Y. Simard, E. Säckinger, Learning prototype models for tangent distance, in: G. Tesauro, D.S. Touretzky, T.K. Leen (Eds.), Advances in Neural Information Processing Systems, Vol. 7, MIT Press, Cambridge, MA, 1995, pp. 999–1006.

[4] G.E. Hinton, P. Dayan, M. Revow, Modeling the manifold of images of handwritten digits, IEEE Trans. Neural Networks 8 (1) (1997) 65–74.

[5] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, K. Torkkola, LVQ_PAK: The Learning Vector Quantization program package, Technical Report A30, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, January 1996, http://www.cis.hut.fi/nnrc/nnrc-programs.html.

[6] L. Mason, P. Bartlett, J. Baxter, Improved generalization through explicit optimization of margins, Technical Report, Deparment of Systems Engineering, Australian National University, 1998.

[7] R. Schapire, Theoretical views of boosting, in: Computational Learning Theory: Proc. 4th European Conference, EuroCOLT'99, Dortmund, Germany, 1999.

[8] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, Boosting the margin: A new explanation for the effectiveness of voting methods, Ann. Statist. 26 (5) (1998).

[9] H. Schwenk, M. Milgram, Learning discriminant tangent models for handwritten character recognition, in: Proc. International Conference on Artificial Neural Networks, Springer, Berlin, 1995, pp. 985–988.

[10] H. Schwenk, M. Milgram, Transformation invariant autoassociation with application to handwritten character recognition, in: G. Tesauro, D.S. Touretzky, T.K. Leen (Eds.), Advances in Neural Information Processing Systems, Vol. 7, MIT Press, Cambridge, MA, 1995, pp. 991–998.

[11] P.Y. Simard, Y. LeCun, J. Denker, Efficient pattern recognition using a new transformation distance, in: S.J. Hanson, J.D. Cowan, C.L. Giles (Eds.), Advances in Neural Information Processing Systems, Vol. 5, Morgan Kaufmann, San Mateo, CA, 1993, pp. 50–58.

[12] P.Y. Simard, Efficient computation of complex distance metrics using hierarchical filtering, in: J.D. Cowan, G. Tesauro, J. Alspector (Eds.), Advances in Neural Information Processing Systems, Vol. 6, Morgan Kaufmann, San Francisco, CA, 1994, pp. 168–175.

[13] D. Sona, A. Sperduti, A. Starita, A constructive learning algorithm for discriminant tangent models, in: M.C. Mozer, M.I. Jordan, T. Petsche (Eds.), Advances in Neural Information Processing Systems, Vol. 9, MIT Press, Cambridge, MA, 1997, pp. 786–792.

[14] D. Sona, A. Sperduti, A. Starita, Discriminant pattern recognition using transformation invariant neurons, Neural Comput. 12 (6) (2000).

[15] A. Sperduti, D.G. Stork, A rapid graph-based method for arbitrary transformation-invariant pattern classification, in: G. Tesauro, D.S. Touretzky, T.K. Leen (Eds.), Advances in Neural Information Processing Systems, Vol. 7, MIT Press, Cambridge, MA, 1995, pp. 665–672.

[16] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.